IMPLEMENTASI KOMPUTASI AWAN MENGGUNAKAN TEKNOLOGI GOOGLE APP ENGINE (GAE) DAN AMAZON WEB SERVICES (AWS)

Adi Nugroho, ST, MMSI[1], Dr Techn Khabib Mustofa, SSi, MKom[2]

ABSTRAK

Interoperabilitas, dalam arti cara bagaimana suatu sistem yang memiliki *platform* perangkat keras dan perangkat lunak tertentu dapat berkomunikasi dengan sistem-sistem yang memiliki *platform* yang berbeda, mungkin merupakan bagian dari 'masa lalu'. Di masa-masa yang akan datang, interoperabilitas yang selama ini ditangani secara manual oleh organisasi-organisasi/perusahaan-perusahaan akan ditangani langsung oleh vendor-vendor penyedia komputasi awan (*cloud computing*) yang memang memiliki sumberdaya-sumberdaya manusia (analis sistem, pemrogram, pakar jaringan), perangkat keras (komputer-komputer server yang berjumlah sangat banyak dan berkemampuan raksasa), serta perangkat lunak (sistem operasi, server aplikasi, server Web) yang memang memenuhi syarat untuk itu. Di masa yang akan datang, untuk mendapatkan layanan-layanan (*service*) dan tempat penyimpanan tertentu, organisasi-organisasi/perusahaan-perusahaan tidak perlu berinvestasi terlalu tinggi untuk menyediakannya sendiri; mereka bisa saja menyewanya dari vendor-vendor komputasi awan yang saat ini mulai bermunculan. Google dan Amazon adalah para pendahulu dari teknologi komputasi awan (*cloud computing*) ini. Melalui tulisan ini, kita tidak akan membahas struktur internal keduanya secara rinci, melainkan kita akan mencoba membahas kelebihan serta kekurangan kedua vendor komputasi awan ini dari sudutpandang para manajer di bidang Teknologi Informasi yang akan melakukan investasi yang bermanfaat bagi organisasi/perusahaannya.

Kata kunci : *Cloud Computing, Google App Engine, Amazon Web Service.*

[1] Staf pengajar di FTI – UKSW (Fakultas Teknologi Informasi – Universitas Kristen Satya Wacana) di Salatiga – Jawa Tengah. Mahasiswa program S3 di FMIPA - UGM (Fakultas Matematika dan Ilmu Pengetahuan Alam – Universitas Gadjah Mada) di Jogyakarta.

[2] Staf pengajar program S3 di FMIPA - UGM (Fakultas Matematika dan Ilmu Pengetahuan Alam – Universitas Gadjah Mada) di Jogyakarta.

PENDAHULUAN : KOMPUTASI AWAN (*CLOUD COMPUTING*)

Komputasi awan (*cloud computing*) pada dasarnya merupakan komputasi masa depan. Dengan difasilitasi oleh jaringan komputer global saat ini (Internet), komputasi awan akan semakin populer di masa-masa yang akan datang. Komputasi awan membuat investasi di bidang Teknologi Informasi akan menjadi semakin 'murah', karena layanan-layanan (*services*) tertentu bisa diperoleh dari vendor yang menyediakannya (SaaS-*Software as a Service*) tanpa organisasi/perusahaan harus mengembangkannya sendiri. Juga, dalam beberapa kasus, organisasi-organisasi/perusahaan-perusahaan tidak perlu lagi menanamkan investasinya dalam bentuk perangkat-perangkat keras serta sistem-sistem operasi (termasuk di dalamnya server-server aplikasi dan server Web) yang mahal harganya, karena organisasi-organisasi/perusahaan-perusahaan cukup menyewanya saja dari berbagai vendor komputasi awan yang ada (dalam kasus seperti yang disebutkan terakhir ini, sebagian orang menyebutnya sebagai PaaS-*Platform as a Service*). Dalam kebanyakan kasus, organisasi-organisasi/perusahaan-perusahaan yang menggunakan fasilitas komputasi awan pada dasarnya tidak perlu lagi mengetahui dimana sesungguhnya layanan yang diperlukannya dilakukan/diproses, serta tidak perlu tahu lagi dimana data mereka disimpan, karena semuanya sudah diatur oleh vendor komputasi awan. Sementara spesifikasi-spesifikasi secara internalnya relatif tidak sama untuk masing-masing vendor, komputasi awan dapat kita pikirkan sebagai sumberdaya-sumberdaya perangkat keras (komputer-komputer, hardisk-hardisk, dan jaringan-jaringan komputer) dan perangkat lunak yang bersifat koheren, berukuran raksasa, dan dapat diakses secara meluas.

Google dan Amazon merupakan para pendahulu dari konsep komputasi awan ini dan diikuti oleh Salesforce. Setelah itu 'pemain besar' seperti Microsoft (dengan fasilitasnya yang dinamakan sebagai Microsoft Azure) juga ikut meramaikannya. Pada dasarnya, pusat-pusat pemrosesan dan pusat-pusat data (*data center*) untuk komputasi awan bisa berada di bagian mana saja di dunia ini dan terdiri dari berbagai jenis komputer (dari komputer besar [*mainframe*], komputer mini, hingga komputer-komputer pribadi [PC-*Personal Computer*]) yang berbeda, sehingga (dalam kebanyakan kasus) teknologi pemrosesan yang memungkinkan berbagai jenis komputer yang berbeda itu (dengan sistem operasi dan *platform*-nya yang beragam) dapat saling berkomunikasi adalah teknologi *Web Service* dan/atau teknologi-teknologi berkaitan dengan interoperabilitas antarsistem lainnya.

Teknologi *Web Service* pada dasarnya merupakan teknologi interoperabilitas yang terkini. Pemain-pemain besar seperti Oracle dengan bahasa pemrograman Java (dengan teknologi JAX-WS [*Java API for XML Processing*] dan JAX-RS [*Java API for REstful Services*]-nya) serta Microsoft (dengan teknologi .NET Web Service-nya), juga teknologi-teknologi dari PHP, Ruby, dan sebagainya, semua sepakat untuk membuat standarisasi pesan (*message*) dan pendefinisian layanan antarsistem mereka yang berbeda, sehingga –minimal secara teoritis- masing-masing sistem yang dikembangkan menggunakan bahasa pemrograman tertentu dapat dimanfaatkan layanan-layanan (*services*)-nya oleh sistem-sistem lain yang dikembangkan menggunakan bahasa-bahasa pemrograman yang berbeda. Standarisasi pesan itu adalah dalam bentuk berkas **XML (*eXtensible Markup Language*)** yang memiliki 'dialek' tertentu, yaitu **SOAP (*Simple Object Access Protocol*)** dan **REST (*REpresentational State Transfer*)** untuk standarisasi pesan-pesan (*message*) yang mengalir antarsistem dan **WSDL (*Web Service Description Language*)** untuk standarisasi layanan (*service*) yang disediakan oleh suatu komponen perangkat lunak tertentu sedemikian rupa sehingga layanan (*service*) itu bisa dimanfaatkan oleh komponen-komponen perangkat lunak atau oleh sistem/aplikasi yang lainnya. Dalam hal ini, teknologi *Web Service* inilah yang memungkinkan terjadinya/terbentuknya komputasi awan. Melalui tulisan ini, kita tidak akan membahas bagaimana

secara internal komputasi awan itu diimplementasikan, melainkan kita akan membahas komputasi awan dari sudutpandang pengguna (baca: mereka yang memiliki wewenang untuk menentukan kebijakan strategis Teknologi Informasi di organisasi/perusahaan tertentu). Kita, melalui tulisan ini, akan membahas secara sepintas implementasi layanan-layanan (*services*) yang disediakan oleh **Google** (dalam bentuk **Google App Engine/GAE**) dan **Amazon** (dalam bentuk **Amazon Web Services/AWS**), kemudian akan melihat kelebihan dan kekurangannya masing-masing.

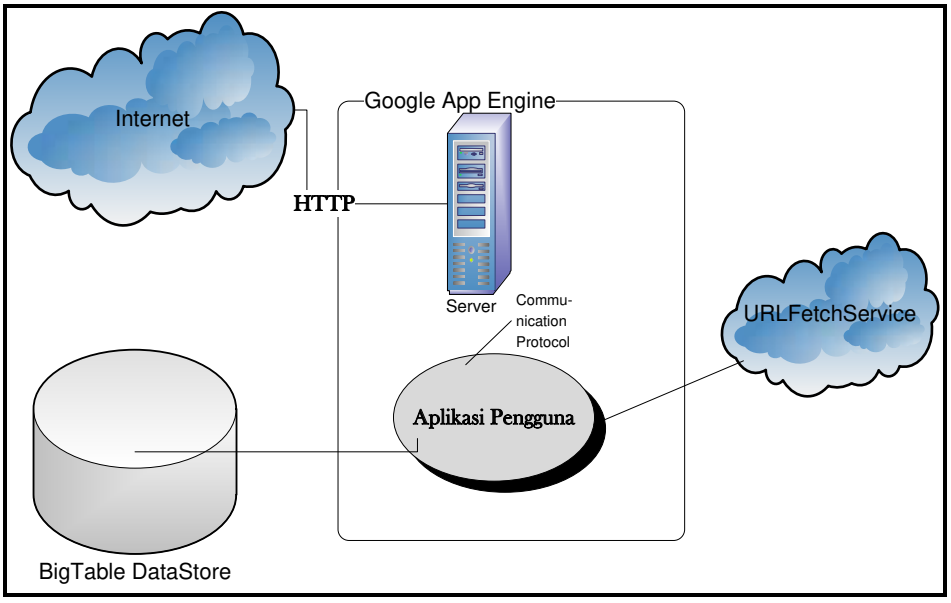
IMPLEMENTASI KOMPUTASI AWAN MENGGUNAKAN TEKNOLOGI GOOGLE APP ENGINE (GAE)

Komputasi awan (*cloud computing*) menurut definisi Google adalah pusat data yang ada di luar komputer aplikasi klien. Tujuan utama dari Google berkontribusi dalam komputasi awan adalah untuk memanfaatkan sumberdaya komputer yang dimilikinya dalam jumlah yang sangat besar yang berjumlah jutaan komputer dan tersebar di seluruh dunia serta untuk mempercepat operasi-operasi yang dilakukan oleh aplikasi-aplikasi Web saat ini. Dalam hal ini, komputasi awan yang dikembangkan oleh Google pada dasarnya berada di seputar jaringan komputer raksasa yang bekerja untuk aplikasi-aplikasi utamanya seperti Google Search dan Google Mail (GMail).

Tabel 1
Biaya Pemanfaatan Google App Engine

Resource	Unit	Unit Cost
Outgoing bandwith	Gigabyte	$0.12
Incoming bandwith	Gigabyte	$0.10
CPU time	CPU hours	$0.10
Stored data	Gigabyte permonth	$0.15
High replication storage	Gygabyte permonth	$0.45
Recipients Emailed	Recipients	$0.0001
Always on	N/A (daily)	$0.30

Saat ini pihak Google melakukan pemantauan terhadap seluruh jaringan komputer yang dimilikinya serta juga melakukan pemantauan terhadap aplikasi-aplikasi yang berjalan di atasnya. Dengan demikian, saat seorang pengguna memanfaatkan layanan komputasi awannya, Google dapat memantau lama waktu kerja CPU (*Central Processing Unit*) yang dilakukan untuk melakukan pemrosesan layanan itu, berapa kapasitas memori yang digunakannya, serta dapat menghitung kapasitas ruang penyimpanan (hardisk) yang digunakannya, sehingga kelak dapat menarik biaya sejumlah US$ tertentu pada penggunanya (perhatikan Tabel 1). Meski demikian, tidak sebarang aplikasi dapat berjalan di komputer-komputer milik Google yang membentuk jaringan komputasi awan itu. Agar dapat berjalan dengan baik, aplikasi-aplikasi harus dikembangkan menggunakan 'aturan-aturan' yang ditetapkan oleh Google. Aplikasi-aplikasi seperti itu dinamakan sebagai **Google App Engine (GAE)**, dimana saat ini aplikasi-aplikasi jenis ini bisa ditulis dengan cara yang relatif mudah oleh para pemrogram komputer yang memahami bahasa pemrograman Java dan Phyton.



Gambar 1
Arsitektur Aplikasi Google App Engine Secara Umum

Arsitektur sistem untuk **Google App Engine (GAE)** ini secara umum diperlihatkan dalam Gambar 1 di atas. Dalam hal ini, **Apps Engine Request** yang dimilikinya menentukan jumlah permintaan (*request*) yang bisa ditangani oleh aplikasi. Jika kita menggunakan fasilitas berbayar, aplikasi kita diijinkan untuk menerima 500 permintaan perdetik (!) (suatu jumlah yang pada umumnya sudah sangat memadai). Demikian juga dengan **DataStore** yang disediakan. Kita bisa menggunakan fasilitas tidak berbayar (yang dapat digunakan untuk melakukan penyimpanan data terbatas sebanyak 1 GB/hari) atau berbayar (data yang disimpan tidak dibatasi) (Dalam hal ini, meskipun pengguna sesungguhnya tidak perlu secara rinci mendalaminya, kita perlu tau bahwa Google App Engine menggunakan sistem basis data non-relasional **BigTable** untuk data yang diletakkan di dalam sistemnya). Selanjutnya, **Apps Engine Request** juga, dengan komponen **URL Fetch Service**-nya, dapat berkomunikasi dengan aplikasi-aplikasi lainnya atau mengakses sumberdaya lainnya yang ada di Web dengan menggunakan URL (*Uniform Resource Locator*) yang dimiliki aplikasi/sumberdaya itu. Suatu aplikasi dapat menggunakan layanan ini untuk mengirim/menerima permintaan HTTP (*Hypertext Transport Protocol*) dan HTTPS (*Hypertext Transport Protocol Secure*) (tentu saja jumlahnya sesuai dengan jumlah dana dalam US$ yang kita tanamkan).

Untuk mengembangkan aplikasi Google App Engine, pengembang yang memiliki preferensi yang tinggi terhadap bahasa pemrograman **Java** bisa menggunakan XMPP (*Extensible Messaging and Presence Protocol*) yang merupakan versi 1.2.5 dari **Java SDK for App Engine**. Layanan ini memungkinkan aplikasi **Apps Engine** berinteraksi dengan layanan-layanan XMPP lainnya seperti (menyebutkan suatu contoh) Google Talk. Mengembangkan aplikasi menggunakan **Google App Engine for Java (GAE/J)** mirip dengan pengembangan aplikasi *Enterprise* menggunakan Java, kecuali pengembang tidak perlu terlalu memikirkan tentang jaringan komputer yang digunakan, perangkat keras, sistem operasi, basis data, atau server aplikasinya. Menggunakan GAE/J, pemrogram aplikasi bisa langsung berinovasi dan mengembangkan aplikasi, tanpa harus melakukan hal-hal yang terlalu teknis seperti melakukan pengaturan-pengaturan sistem operasi dan melakukan konfigurasi-

konfigurasi basis data. GAE/J menyediakan antarmuka-antarmuka (*interface*) yang serupa dengan JVM (*Java Virtual Machine*) versi 6 dan Java Servlet, dan juga mendukung teknologi-teknologi Java yang umum seperti (hanya menyebutkan beberapa di antaranya) JDO (*Java Database Object*), JPA (*Java Persistence API*), JavaMail, dan sebagainya. Aplikasi-aplikasi GAE/J dapat dikembangkan menggunakan IDE **Eclipse** dan **Google Plugin for Eclipse** yang menyediakan bagi kita server pengembangan yang bersifat lokal, sehingga pengembang bisa mengembangkan aplikasi dari awal hingga akhir, sebelum mengunggahnya (*upload*) menjadi **Java Google App Engine.**

Di samping penggunaan bahasa pemrograman Java, Google App Engine juga (yang sangat direkomendasikan) dapat dikembangkan menggunakan bahasa pemrograman **Phyton** dengan/tanpa *platform* **Django**-nya. Jika pengguna menginginkan 'cara cepat' untuk mengembangkan aplikasi-aplikasi Web untuk dijadikan aplikasi-aplikasi Google App Engine, pengguna juga bisa menggunakan kakas (*tool*) yang dinamakan sebagai **GWT (*Google Web Toolkit*)** atau menggunakan *framework* **Spring**. GWT memungkinkan pengguna untuk mengembangkan aplikasi-aplikasi Web yang responsif seperti layaknya aplikasi-aplikasi *desktop* namun sangat kaya fitur dimana hal ini dapat dicapai dengan cara mengadopsi konsep RIA (*Rich Internet Applications*). Meski demikian, dibandingkan dengan aplikasi-aplikasi Web biasa, GWT memungkinkan aplikasi-aplikasi Web dijalankan dalam konteks komputasi awan (bukan sekedar aplikasi Web 'biasa').

IMPLEMENTASI KOMPUTASI AWAN MENGGUNAKAN TEKNOLOGI AMAZON WEB SERVICES (AWS)

Amazon.com sebelumnya lebih terkenal dengan toko buku *on-line*-nya. Meski demikian, beberapa tahun yang lalu (sekitar tahun 2005), Amazon mengembangkan dirinya menjadi **AWS (*Amazon Web Service*)** yang menyediakan layanan komputasi awan, dimana setiap fungsi yang ada di dalamnya bisa diakses dengan panggilan *Web Service*. Protokol-protokol *Web Service* yang digunakan adalah SOAP dan REST. Layanan-layanan di AWS dapat dimanfaatkan berdasarkan 1) Waktunya (waktu penggunaan CPU), 2) Volume (jumlah data yang ditransfer), 3) Perhitungan (jumlah antrian pesan [*message*]), serta 4) Waktu dan ruang (penggunaan ruang hardisk dalam periode waktu tertentu). Secara infrastruktur internal, AWS memiliki komponen-komponen sebagai berikut.

- **Amazon S3 (*Simple Storage Service*)**. Digunakan untuk menyimpan data untuk penggunaan pribadi maupun umum. Dalam hal ini, ada 3 lokasi yang memungkinkan pemanfaatannya, yaitu di Amerika Serikat (termasuk California Utara), Eropa, serta Asia.
- **Amazon Cloud Front**. Digunakan untuk mendukung Amazon S3 agar bisa bekerja dengan lebih baik dan lebih cepat.
- **Amazon SQS (*Simple Queue Service*)**. Digunakan untuk mendukung tercapainya pemrosesan AWS yang cepat dan tidak pernah mengalami kegagalan.
- **Amazon SimpleDB**. Digunakan untuk menyimpan data yang bersifat semi-terstruktur. Basis data yang digunakan (SimpleDB) tidak bersifat relasional, melainkan menyimpan data dalam bentuk pasangan nama/nilai (*name/value*) yang mirip dengan struktur denormalisasi pada sistem basis data relasional, demi meningkatkan kinerja *query*.
- **Amazon RDS (*Relational Database Service*)**. Digunakan untuk mengelola data yang disimpan dalam sistem basis data MySQL.

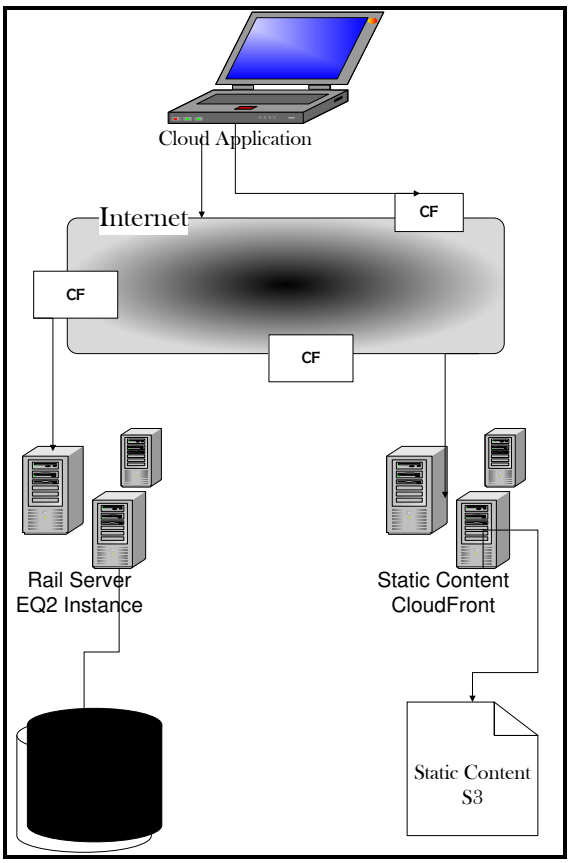
- **Amazon EC2 (*Elastic Compute Cloud*).** Digunakan sebagai infrastruktur (kapasitas pemrosesan, memori, dan ruang hardisk) yang menyediakan layanan (*service*) yang dibutuhkan oleh para pengguna.

Tabel 2
Jenis-jenis *Instance* Amazon EC2

Name	CPU Word Size	CPU Virtual Cores	CPU Core Speed (EC2 Compute Units)	RAM	Local Disk	Cost/Hour
Email	32-bit	1	1	1.7 GB	160 GB	$0.085
Large	64-bit	2	2	7.5 GB	850 GB	$0.34
Extra Large	64-bit	4	2	15 GB	1690 GB	$0.68
High-CPU Medium	32-bit	2	2.5	1.7 GB	350 GB	$0.17
High-CPU Extra Large	64-bit	8	2.5	7 GB	1690 GB	$0.68
High Memory Extra Large	64-bit	2	3.25	17.1 GB	420 GB	$0.50
High Memory Double Extra Large	64-bit	2	3.25	34.2 GB	850 GB	$1.20
High Memory Quadruple Extra Large	64-bit	8	3.25	68.4 GB	1690 GB	$2.40
Cluster Computer	64-bit	8 (2 processors each with 4 cores)	2.5	23 GB	1690 GB	$1.60

Konsep yang sangat penting dalam Amazon Web Service (AWS) adalah *instance* (Tabel 2). Menggunakan suatu teknik yang dinamakan sebagai virtualisasi, para pengguna bisa melakukan pengembangan aplikasinya di atas berbagai perangkat keras dengan cara yang serupa dengan saat pengembangan aplikasi dilakukan pada sebuah mesin tunggal. Perangkat lunak virtualisasi memastikan bahwa masing-masing *instance* secara logika dapat saling berbagi waktu kerja CPU (*Central Processing Unit*) dan berbagi ruang memori dengan cara yang benar tanpa saling berinterferensi satu dengan lainnya. Dapat kita lihat melalui Tabel 2, setiap *instance* dengan waktu kerja tertentu serta alokasi RAM (*Random Access Memory*) tertentu mensyaratkan investasi sejumlah US$ tertentu. Sebagai catatan tambahan, Tabel 1 di atas memperlihatkan investasi yang diperlukan untuk mengembangkan aplikasi komputasi awan di Amerika Serikat dengan *platform* sistem operasi

Linux. Untuk sistem-sistem operasi lain serta untuk kawasan-kawasan lain, Amazon menyediakan perhitungan-perhitungan lain.



Gambar 2
Arsitektur Aplikasi Amazon EC2

Pengembangan sistem dalam Amazon Web Service (AWS) –tepatnya Amazon EC2 (*Elastic Compute Cloud*)- pada umumnya dilakukan berdasarkan metoda **CBD (*Component Based Development*)** dimana komponen-komponen yang ada dapat saling berkomunikasi dan bertukar data menggunakan suatu komponen perangkat lunak AWS yang bernama Amazon Simple Queue Service (SQS). Selanjutnya, sistem yang besar, yang menggunakan ruang penyimpanan berukuran besar, dapat melandaskan dirinya pada fasilitas yang ada di dalam Amazon Simple Storage Service (S3). Dalam hal ini, berlainan dengan pendekatan yang dilakukan oleh Google dengan sistem basis data non-relasionalnya (BigTable), Amazon menggunakan berbagai sistem basis data yang bertindak sebagai layanan (*service*) bagi komponen-komponen yang membutuhkan data. Sistem basis data yang digunakan Amazon Web Service (AWS) berupa baik sistem data relasional (RDS-*Relational Database Server*) maupun non-relasional (SimpleDB) yang dimotori oleh sistem basis data non-relasional Dynamo. Dalam semua hal ini, untuk mengoptimalkan layanannya, Amazon Web Service (AWS) menggunakan perhitungan utilitas perangkat keras dan menyeimbangkannya menggunakan suatu fasilitas yang dinamakan sebagai ELB (*Elastic Load Balancing*). Untuk melakukan pengaturan komponen-komponen agar mengelompok berdasarkan apa yang dilakukannya serta berdasarkan utilitasnya, Amazon menggunakan apa yang dinamakan sebagai Amazon Autoscaling. Sementara untuk melakukan pemantauan terhadap CPU-CPU, lalulintas jaringan, serta penggunaan hardisk yang

melintas jaringan komputer server dalam sistem komputer awan (*cloud computing*) yang dimiliki Amazon digunakan suatu komponen perangkat lunak dalam sistem Amazon Web Services (AWS) yang dinamakan sebagai Amazon Cloud Watch.

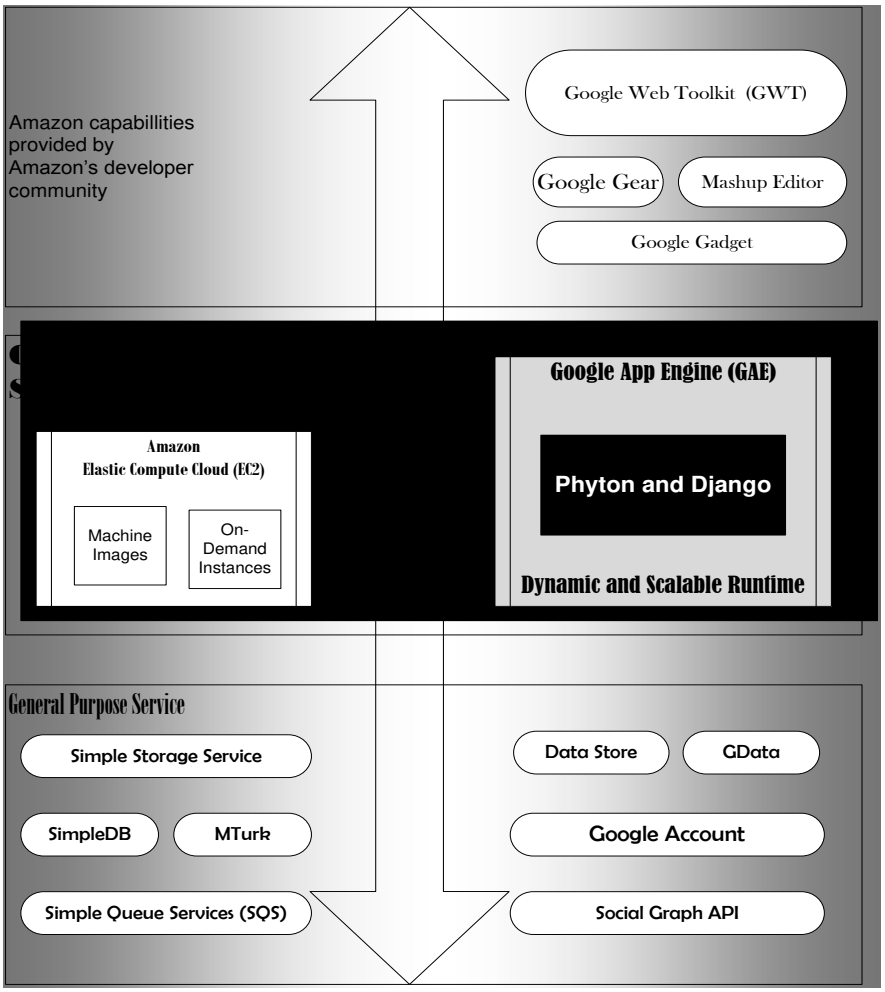
Pengembangan sistem dalam Amazon Web Service (AWS) dilakukan pertama kali dengan membuat **AWS Account** yang meski untuk penggunaan awal gratis, tetapi menuntut penggunanya memiliki dukungan pembiayaan melalui kartu kredit tertentu (paling tidak memiliki akun kartu kredit tertentu). Kemudian, kita bisa melakukan pengelolaan dan pengaturan aplikasi yang dikembangkan menggunakan **AWS Management Console**. Selanjutnya, berkaitan dengan aplikasi yang kita kembangkan, kita perlu memilih arsitektur sistem yang akan digunakan, seperti yang terlihat dalam Gambar 2. Jika organisasi/perusahaan memilih **Rail Server**, Amazon akan menyediakan bagi organisasi/perusahaan *framework* Rail, server Web Apache, serta fasilitas Passenger yang bermanfaat untuk melakukan *deployment* aplikasi. Alternatif lainnya (perhatikan juga Gambar 2), kita bisa memilih arsitektur **S3/CloudFront**, dimana pengembang bisa mengembangkan aplikasi yang bersifat 3 *tier* (*data tier, logic tier,* dan *presentation tier*) secara lebih konsisten. Menggunakan alternatif yang terakhir ini, organisasi/perusahaan memiliki kebebasan yang lebih penuh untuk menentukan lokasi-lokasi dimana berkas-berkas (*file*) akan diletakkan, memiliki kebebasan penuh untuk melakukan konfigurasi-konfigurasi server aplikasi, dan sebagainya, yang pada gilirannya akan memberikan fleksibilitas penuh pada aplikasi yang dikembangkan (dengan konsekuensi tingkat kesulitan pengembangan sistem akan lebih tinggi).

ANALISIS TENTANG IMPLEMENTASI KOMPUTASI AWAN MENGGUNAKAN TEKNOLOGI GOOGLE APP ENGINE (GAE) DAN AMAZON WEB SERVICES (AWS)

Pada dasarnya, layanan komputasi awan yang diberikan oleh baik Google maupun Amazon memiliki banyak kesamaan. Baik Google maupun Amazon menawarkan penggunaan fasilitas komputasi awan kepada pengguna berdasarkan waktu kerja CPU, *bandwith*, serta berdasarkan ruang penyimpanan yang diperlukan oleh aplikasi pengguna. Perbedaannya yang berkaitan dengan penggunaan awalnya adalah bahwa Google App Engine (hingga penggunaan waktu kerja CPU sebesar 6,5 jam/hari) tidak mensyaratkan akun khusus (yang berbayar) untuk uji-coba, sementara Amazon EC2 (meskipun juga tidak mensyaratkan pengguna membayar uang sejumlah tertentu pada penggunaan uji coba) mensyaratkan akun khusus dengan harus menyebutkan cara pembayarannya. Dari sudutpandang penggunaan ruang tempat penyimpanan, Google juga lebih 'murah hati' dibandingkan dengan Amazon S3. Penggunaan ruang sampai 1 GB/bulan di Google masih gratis (meskipun di atas itu pengguna harus membayar sejumlah US$ tertentu). Bagaimana pun juga ini masih lebih baik daripada Amazon S3 yang mensyaratkan penggunaan kartu kredit sejak awal penggunaan ruang.

Dari sudut pengembangan aplikasi, Google App Engine mensyaratkan pemahaman bahasa pemrograman Java dan Phyton, sementara Amazon lebih menekankan pada penggunaan bahasa pemrograman PHP dan Ruby (walaupun penggunaan Java secara relatif terbatas juga didukung). Dalam hal ini, mungkin ini salah satu kelemahan dari Google App Engine, yaitu bahwa struktur internal Google lebih banyak dikembangkan menggunakan bahasa pemrograman Phyton dengan *platform* Django (yang saat relatif kurang populer), sehingga agak membatasi pengguna yang menginginkan akses lebih besar terhadap sistem dan yang mengembangkan aplikasinya menggunakan

bahasa-bahasa pemrograman lain, seperti Java, bahasa-bahasa pemrograman dalam keluarga .NET, PHP, dan sebagainya. Dari sudutpandang implementasi sistem basis data, aplikasi-aplikasi Google App Engine tidak memungkinkan pengguna untuk terlalu banyak melakukan kendali atas sistem basis data (sebagian besar kendali atas basis data non-relasional yang digunakan Google App Engine ditangani langsung bukan oleh pengguna melainkan oleh sistem), sementara pada Amazon Web Service, kita lebih mampu mengendalikan sistem basis data apa yang akan pengguna gunakan serta bagaimana caranya menanganinya (dalam hal terakhir ini, jika kita mau, kita bisa menggunakan SimpleDB yang lebih memungkinkan pengguna untuk mengendalikan sistem).



Gambar 3
Perbandingan Implementasi Menggunakan
Amazon Web Service (kiri) dan Google App Engine

Dari sudutpandang kemudahan pengembangan aplikasi (termasuk melakukan *deploy* aplikasi) yang memakan waktu sekitar 30-35% dari keseluruhan waktu yang diperlukan untuk menyelesaikan siklus pengembangan sistem (SDLC-*System Development Life Cycle*), aplikasi Google App Engine (GAE) relatif lebih mudah dikembangkan daripada pengembangan aplikasi Amazon Web Service (AWS),

karena Google menyediakan beragam SDK (*Standard Development Kit*) untuk Windows, Mac, Linux, dan sebagainya. Juga, di sudut aplikasi klien, Google App Engine (GAE) bisa dikembangkan menggunakan kakas-kakas (*tool*) yang relatif lebih mudah (misalnya GWT [*Google Web Toolkit*] untuk pengembangan aplikasi-aplikasi Web dalam konteks komputasi awan) dibandingkan dengan Amazon Web Service (AWS) yang kode-kodenya lebih sulit dan hanya dikembangkan di komunitas-komunitas pengembang aplikasi-aplikasi Amazon Web Service (AWS) tertentu. Konsekuensinya, karena Google App Engine lebih mudah dikembangkan, maka pemeliharaannya juga lebih mudah dilakukan. Meski demikian, karena pengembangan aplikasi Google App Engine lebih mudah dilakukan, dimana hal ini lebih disebabkan oleh banyaknya campur-tangan sistem yang disediakan Google, kendali pengguna pada aplikasi Google App Engine kurang maksimal dibandingkan kendali pengguna pada Amazon Web Service. Dalam hal pengembangan aplikasi ini, Amazon Web Service memiliki keunggulan dibandingkan Google App Engine, dimana pada yang terdahulu kreativitas pengguna sama sekali tidak terbatas (bahkan hingga pada pengendalian atas sistem yang mendasari) dibandingkan yang terakhir yang -karena kendali pengguna atas aplikasinya relatif rendah-kreativitas pengguna relatif menjadi agak terbatas.

Dari sudutpandang teknologi kita memang tidak bisa mengetahui secara mendalam dan rinci teknologi apa yang sebenarnya diimplementasikan oleh baik Google maupun Amazon. Meski demikian, jika kita meminjam prinsip teknologi pengujian 'kotak hitam' (*black box*), yaitu berdasarkan apa yang 'tampak' di luar, kelihatannya Google sedikit lebih unggul. Sebagai contoh, Google dapat menerima permintaan (*request*) akan aplikasi dalam jumlah yang lebih tinggi dibandingkan Amazon (6,5 jam-penggunaan CPU/hari dan 1,3 juta permintaan/hari pada Google App Engine dan pada Amazon Web Service parameter-parameter itu bergantung pada jumlah dana yang organisasi/perusahaan investasikan). Demikian juga dengan kemampuan menangani data masuk/data keluar, dimana server-server Google memiliki kemampuan yang lebih tinggi dibandingkan server-server Amazon Web Service (*outgoing bandwth* dan *incoming bandwith* pada Google adalah sebesar 1 GigaByte/hari dan pada Amazon 1 GigaByte/bulan). Juga, yang penting bagi pengguna awal, Google menyediakan tempat penyimpanan data sebesar 1 Gigabyte/bulan secara cuma-cuma sementara Amazon mensyaratkan investasi dalam bentuk dana sebesar US$ tertentu (setidaknya dalam bentuk pemberian nomor akun untuk kartu kredit tertentu), berapa pun besarnya tempat penyimpanan data yang digunakan.

KESIMPULAN

Sudah kita lihat di atas bahwa komputasi awan (*cloud computing*) merupakan sarana pengembangan aplikasi yang sangat penting di masa yang akan datang, karena menggunakan komputasi awan ini pengguna tidak perlu lagi memikirkan infrastruktur yang mendasari suatu aplikasi yang biasanya memerlukan investasi dalam bentuk dana yang sangat besar. Selain itu, dari sudutpandang pengguna, komputasi awan memungkinkan pengguna mengembangkan aplikasi secara terintegrasi mulai dari perancangan sistem, perancangan antarmuka pengguna (*user interface*), perancangan basis data, pemrograman, dan hal hal-hal yang bersifat teknis lainnya. Selanjutnya, jika kita melihat pembahasan sebelumnya ada beberapa hal yang dapat kita simpulkan berkaitan dengan implementasi Google App Engine (GAE) dan Amazon Web Service (AWS). Kesimpulan yang dibuat secara umum itu adalah sebagai berikut.

Google App Engine (GAE).

Kelebihan.

1. Aplikasi dapat dengan mudah dan cepat dikembangkan menggunakan bahasa pemrograman Java dan/atau Phyton dan tidak memerlukan usaha pemeliharaan yang ekstra sulit.
2. Biaya penggunaan relatif murah (bahkan gratis untuk aplikasi-aplikasi kecil) dan, jika kuota lebih diinginkan, penambahan kuota dapat dilakukan dengan cara yang sangat masuk akal.

Kekurangan.

1. Aplikasi-aplikasi tertentu, terutama yang memerlukan akses penuh ke sistem yang mendasari, relatif sukar untuk dikembangkan.
2. Sistem yang terkendali penuh tidak memungkinkan pustaka-pustaka (*library*) dan *framework-framework* tertentu digunakan oleh aplikasi-aplikasi GAE.
3. Tidak mendukung penggunaan sistem basis data relasional.

Amazon Web Services (AWS).

Kelebihan.

1. Aplikasi-aplikasi AWS yang ditulis menggunakan bahasa-bahasa pemrograman PHP, Ruby, serta Java, dapat dikembangkan dengan cara yang sangat fleksibel karena pengguna memiliki kendali penuh pada sistem yang mendasari.
2. Struktur pembiayaan sederhana.
3. Bisa menggunakan sistem basis data (relasional maupun non-relasional) apa saja yang dibutuhkan oleh pengguna.
4. Jika pengguna mau, pengguna bisa saja menggunakan/menambahkan server-server yang berada di luar Amazon Web Service.

Kekurangan.

1. Kurva belajar yang terjal (relatif sulit untuk mempelajari pengembangan aplikasi-aplikasi di atas Amazon Web Service dibandingkan di atas Google App Engine).
2. Memerlukan waktu yang relatif lebih lama untuk mengembangkan aplikasi (bahkan untuk aplikasi-aplikasi yang relatif sederhana).

Pemilihan organisasi/perusahaan saat akan menggunakan Google App Engine (GAE) atau Amazon Web Service (AWS) sebagai strategi perusahaan untuk mengimplementasikan aplikasi komputasi awannya pada dasarnya sangat bersifat kasuistik. Pemilihan bisa dilakukan berdasarkan sumberdaya manusia yang dimiliki organisasi/perusahaan (terutama ketrampilan teknis para pemrogram komputer yang dimiliki oleh organisasi/perusahaan) serta jenis aplikasi seperti apa yang akan dikembangkan (apakah menuntut kendali penuh pada sistem yang mendasari atau tidak). Pertimbangan lainnya juga bisa digunakan, yaitu dana investasi. Secara umum, investasi dana yang diperlukan relatif berimbang,

tetapi untuk aplikasi-aplikasi yang berukuran relatif kecil, kelihatannya penggunaan Google App Engine lebih ekonomis. Untuk aplikasi-aplikasi komputasi awan yang sangat besar dan kompleks, kelihatannya penggunaan Amazon Web Service lebih direkomendasikan karena aplikasi-aplikasi bisa dikembangkan dengan akses penuh ke sistem yang mendasari.

DAFTAR PUSTAKA

Ahuja, Sanjay, Jee-Lon Yang, 2010. *Performance Evaluation of Java Web Service : A Developer's Perspective*. Communications and Networks, 2010, 2, 200-206.www.SciRP.org/journal/cn.

Azeez, Afkham. *Autoscaling Web Services on Amazon EC2.* Department of Computer and Engineering-University of Moratuwa-Srilanka.

Babcock, Charles, 2010. *Management Strategies for the Cloud Revolutions : How Cloud Computing Is Transforming Businesses and Why You Can't Afford to Be Left Behind*. McGraw-Hill Companies, New York-USA.

Bar, Jeff, 2010. *Host Your Web Site in the Cloud : Amazon Web Service Makes Easy*. Amazon Web Service-Site Point Pty Ltd., Seattle-USA. www.sitepoint.com.

Blokdijk, Gerrard, 2008. *SaaS Success Secrets*. www.ebooke.org.

Chappell, David, 2008. *Introducing The Azure Services Platform : An Early Look at Windows Azure, .NET Services, SQL Services, and Live Services.* Microsoft Corp. and David Chappell Associates, San Fransisco-California-USA. www.davidchappell.com.

Ciurrana, Eugene. *Developing with Google Apps Engine.* Apress. www.apress.com.

Habeb, Mocky, 2011. *A Developer's Guide to Amazon SimpleDB*. Addison Wesley-Pearson Education, Boston-USA.

Hansen, Mark D., 2007. *SOA Using Java Web Services.* Pearson Education, Inc., Upper-Saddle River-New York.

Hinchcliffe, Dion. *Comparing Amazon's and Google's Platform as Service (PaaS)*. www.zdnet.com.

Kereki, Frederico, 2010. *Building for the Web with Google Web Toolkit 2*. Addison-Wesley Inc.-Pearson Education, Boston-USA.

Lim, Billy, Sri Ram Ajjarapu, Khrisna Kumala, 2004. *Interfacing with Amazon Web Services Using Java and .NET : A Comparative Study*. Journal of Internet Commerce Vol 3(4) 2004. The Haworth Press, Inc., www.haworthpress.com.

Malaher, Tom, 2009. *Cloud Computing and Amazon Web Services*. CJUG.

Nugroho, Adi, 2011. *Implementasi Java Web Service Menggunakan "Big" Web Service dan REST (REpresentational State Transfer*). Dipresentasikan di KNSI (Konferensi Nasional Sistem Informasi), STMIK Potensi Utama, Medan.

Nugroho, Adi, 2010. *Peran Teknologi Komputasi Awan (Cloud Computing) Dalam Pemeliharaan dan Pemulihan Data Kependudukan Pasca Bencana.* Dipresentasikan di SNASTI (Seminar Nasional Teknologi Informasi) di STIKOM Surabaya.

Nakhimovsky, Alexander, Tom Myers, 2004. *Google, Amazon, and Beyond : Creating and Consuming Web Service*. Springer-Verlag, GmBH, Heidelberg, Germany.

Rees, George, 2009. *Cloud Application Architectures.* O'Reilly Media Inc., Sebastopol-USA.

Roche, Kyle, Jeff Douglas, 2009. *Beginning Java Google Apps Engine*. Springer-Verlag, USA.

Teknik pengembangan klien Java untuk Amazon EC2. http://java.sun.com/developer/technicalArticles/WebServices/amazonws/.

Valdes, Ray, 2008. *Google Goes Up Againts Amazon Web Services.* Gartner Inc., Stamford-USA.

Varia, Jinesh, 2010. *Architecting for the Cloud : Best Practises*. Amazon Web Service, Seattle-USA.

Van Fliet, Flavia Paganilli, 2011. *Programming Amazon EC2*. O'Relly Media, Inc., Sebastopol-USA.